blazemeter.com

ASP.NET Login Testing with JMeter | BlazeMeter

Dmitri TikhanskiDmitri Tikhanski is a Contributing Writer to the BlazeMeter blog.

6-8 minutes

ASP.NET is a unified web development model which allows developers to quickly build enterprise level dynamic web applications. It was released by Microsoft in the early 2000s and became an industry-level standard. About 20% of web servers around the globe are powered by Microsoft IIS and this usually means that they are running ASP.NET web applications. If your URL ends with ".aspx" - you're testing an ASP.NET application.

This post will show you how to perform login testing for asp.net applications. However, we'll try to provide a broader view and cover the generic web application login challenge when testing.

The General Approach to Simulating a Login Event

First of all you need to capture the login chain and replicate it using JMeter HTTP Request Samplers. In the majority of cases it will look as follows:

- Open the login page (usually a GET request to a login endpoint)
- Detect and record cookies

- Detect and record static parameters
- Detect and record dynamic parameters
- Perform login (usually a POST request to a login endpoint)
- Provide valid credentials
- Provide mandatory static parameters
- Provide mandatory dynamic parameters
- Have proper cookies
- In some cases get the relevant authentication and other headers

To build the initial login script we recommend using one of the following techniques:

1. Record both requests via JMeter embedded HTTP Proxy Server
2. Use the BlazeMeter Google Chrome extension which works right in browser and doesn't require additional proxies or SSL certificate setup

Alternatively, you can build requests manually but using the record-and-replay capabilities of JMeter and/or the Chrome extension will result in additional productivity of your testing activity.

In general, the next steps required to construct successful login samplers combinations are:

1. Record a login sequence 2 or more times
2. Inspect requests being sent for static and dynamic parameters
3. Perform correlation: extract dynamic parameters from the 1st sampler response, save them to JMeter variables, add the variables to the next request

4. Optionally perform an assertion to ensure that the login took place and was successful

ASP.NET Specifics

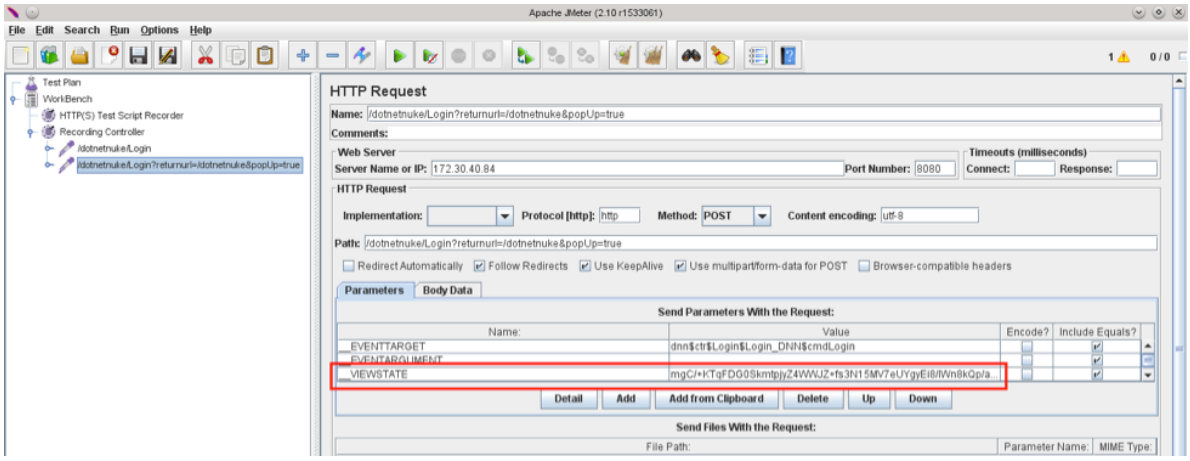
The most common dynamic request parameter for ASP.NET pages is viewstate. Viewstate is an ASP.NET page-level state management mechanism. It stands for the method which is being used to preserve page and controls between round-trips by holding the state of an ASP web form across postbacks. In short, the current state of the page, controls, values, etc. is being converted to a Base64-encoded string and is bi-directionally passed between client and server.

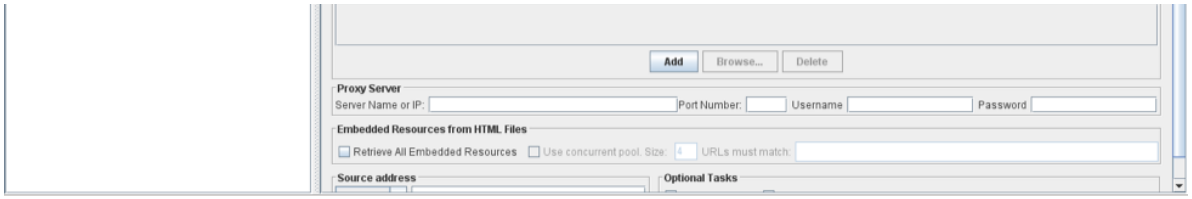
In the web page it's usually represented as a hidden input with an id of "__VIEWSTATE"

<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="YZ5NEj….=" />

In this example, we'll be using DotNetNuke Open Source Community Edition - ASP.NET Web Content Management System deployment for JMeter Log In demonstration.

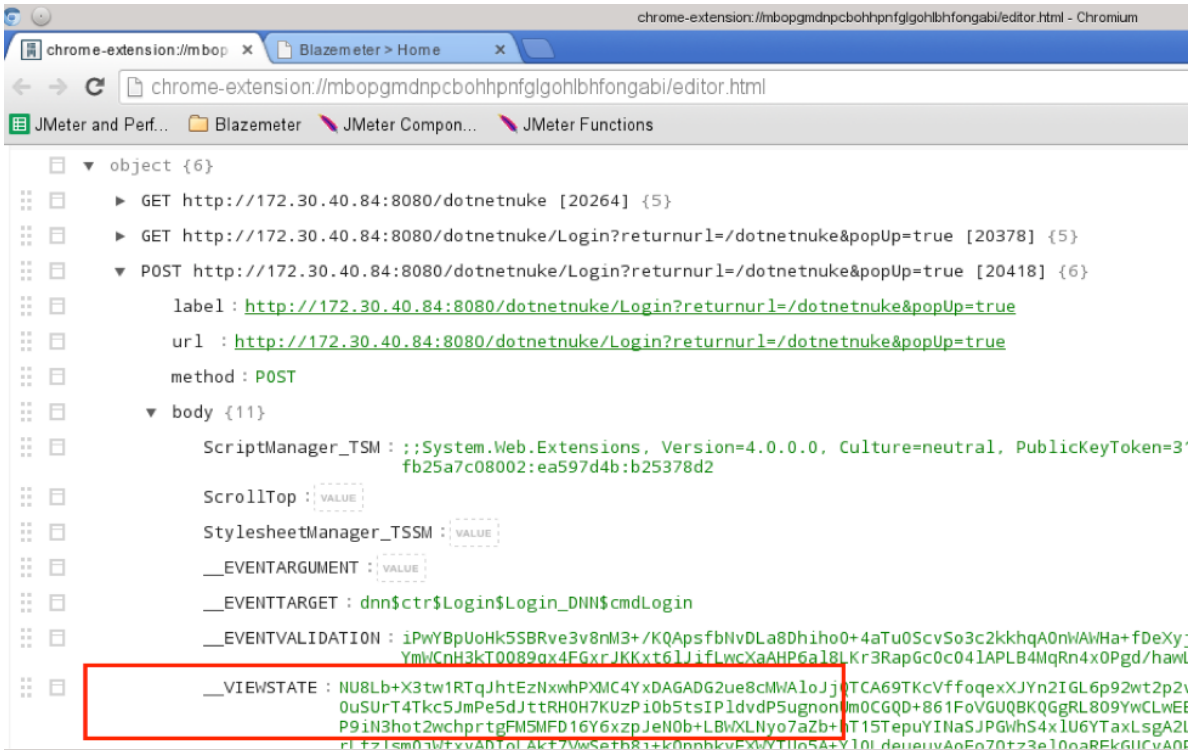
Record the first request using JMeter HTTP Proxy server and mention __VIEWSTATE parameter value:





We see something like **“mgC/+KTqFDG0SkmtpjyZ4WWJ……”**

Let’s retry the login request using the [BlazeMeter Google Chrome Extension](). Go to “Edit the recording before upload” and expand the relevant POST request to “/Login” path:



This time we observe that “__VIEWSTATE” starts from “NU*LB+X3…”

So it’s obvious that we need to somehow obtain the __VIEWSTATE value from the first request response (Get Login Page) and pass it as a parameter to the second request (Perform Login).

JMeter offers several Post Processors designed to extract data from server response and store it as JMeter variables. We’ve already highlighted some of them, including:

- Regular Expression Extractor - which is the most popular and the less resource consuming option
- XPath Extractor - (the post also covers the JSON Path Extractor)

In this example, we'll use CSS/JQuery Extractor. We're doing this since this extractor is only covered in official documentation, and hopefully we'll be able to provide some valuable information here.

So we have a hidden input with "name" and "id" attributes of "__VIEWSTATE" and we're interested in its "value" bit.

To extract the hidden "__VIEWSTATE" input value from the first request, which is the Get Login Page, add the CSS/JQuery Extractor as a child of the request and populate its fields as follows:

- Implementation drop-down: choose one of
- JSOUP
- JODD
- Reference name: anything meaningful, this will be a variable holding the extraction result. Let's put "viewstate" here.
- CSS/JQuery Expression: depending on the Implementation drop down. Any of "input[id=__VIEWSTATE]" or "#__VIEWSTATE" should work.
- Attribute: DOM attribute we're interested in. In this case it would be "value"

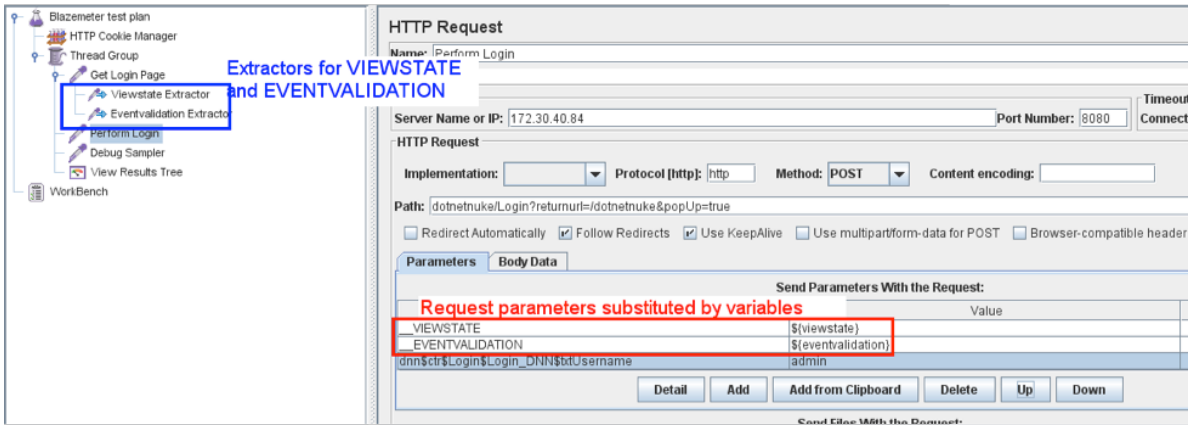
All of the above values should go without quotes. You can also change the extractor name to be self-explanatory, something like "Viewstate Extractor".

CSS/JQuery expressions can be evaluated either right in View

Results Tree Listener or via its combination with Debug Sampler. Refer to How to debug your Apache JMeter script for more information.

Once the “viewstate” variable is being correctly populated by CSS/JQuery extractor you can substitute the relevant POST request parameter which is hard-coded by JMeter Proxy or BlazeMeter plugin with a brand new extracted value.

Still experiencing problems with your login? In our case we have one more dynamic parameter which is called __EVENTVALIDATION which can be handled using the same approach.



Repeat these “correlation” steps for any other dynamic request parameter.

Troubleshooting

We truly hope that you won’t need to scroll down to this section, but just in case, the following few lines can save you loads of time.

1. Make sure that you have added and enabled HTTP Cookie Manager
2. Security for ASP.NET applications in general and Sharepoint in particular can be enhanced by NTLM or Kerberos authentication. If

it's you case refer to the Windows Authentication with Apache JMeter guide

3. If your site is protected by a form of CSRF filter you may need to construct a proper header and add it via the HTTP Header Manager.
4. Anything specific comes up? Just let us know.